Sheet 9 (Interactive graphics programs)

1.

```
// pop-up simple menu demo
#include "stdafx.h"
#include <stdlib.h>
#include <GL/glut.h>
#include <math.h>
#include <string.h>

/* globals */
GLsizei wh = 500, ww = 500; /* initial window size */
GLfloat size = 3.0;   /* half side length of square */

void drawSquare(int x, int y)
{
    y=wh-y;// convert from window coordinates to world coordinates
    // choose a random color
    glColor3ub( (char) rand()%256, (char) rand()%256, (char) rand()%256);
    glBegin(GL_POLYGON);
    glVertex2f(x+size, y+size);
    glVertex2f(x-size, y+size);
    glVertex2f(x-size, y-size);
    glVertex2f(x+size, y-size);
    glEnd();
    glFlush();
}
void demo_menu(int id)
{
    switch(id)
    {
    case 1: exit(0);
        break;
    case 2: size =2 * size;
        break;
    case 3: if(size >1) size = size/2;
        break;
    }
    glutPostRedisplay( );// invoke display function to redraw without the menu
}
void myMouse(int btn, int state, int x, int y)
{
    if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN) drawSquare(x,y);
    if(btn==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)   exit(0);
}
void display()
{

}
void myReshape(GLsizei w, GLsizei h)
{
    /* adjust clipping box */
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, (GLdouble)w, 0.0, (GLdouble)h, -1.0, 1.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    /* adjust viewport and clear */
    glViewport(0,0,w,h);
    glClearColor (1.0, 1.0, 1.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glFlush();
    /* set global size for use by drawing routine */
    ww = w;
    wh = h;
}
void myinit()
{
    glViewport(0,0,ww,wh);
```

```
67      glMatrixMode(GL_PROJECTION);
68      glLoadIdentity();
69      glOrtho(0.0, (GLdouble) ww , 0.0, (GLdouble) wh , -1.0, 1.0);
70      glMatrixMode(GL_MODELVIEW);
71      glClearColor (1.0, 1.0, 1.0, 1.0);
72      glClear(GL_COLOR_BUFFER_BIT);
73      glFlush();
74      glColor3f(0.0,0.0,0.0);
75  }
76  int main(int argc, char **argv)
77  {
78      glutInit(&argc, argv);
79      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
80      glutInitWindowSize(500, 500);
81      glutCreateWindow("Mouse event");
82      glutReshapeFunc(myReshape);
83      glutMouseFunc(myMouse);
84      glutCreateMenu(demo_menu);
85      glutAddMenuEntry("quit",1);
86      glutAddMenuEntry("increase square size", 2);
87      glutAddMenuEntry("decrease square size", 3);
88      glutAttachMenu(GLUT_RIGHT_BUTTON);
89
90      glutDisplayFunc(display);
91      myinit();
92      glutMainLoop();
93  }
```

2. (a)

In an OpenGL application, the application program puts primitives in the GL pipelines to be processed and finally reach the frame buffer ad rasterized pixels. We specify objects using primitives (for example, a sphere is approximated using a set of triangles at the poles and a set of quads in between the two poles). Picking is the process of identifying a primitive or an object in the world model by interacting with the contents of the graphics window on the screen (for example, click to identify a sphere on the screen that contains many other, may be overlapping, objects). OpenGL supports picking but requires some programming.

(b)

In a graphic application that supports interactive modeling, the user, not the programmer, specify his model. For example, The user of the AutoCAD (civil or architect engineer) specifies his model (building). In an interactive modeling application:

- The user can add or remove parts of the model (walls, windows, doors, for examples)
- The user can pick an object to delete, modify, move, etc.
- The user can save/reload his model
- The model must be maintained in a data structures to support interactive modeling

3.

```
// rotating square
#include "stdafx.h"
#include <stdlib.h>
#include <GL/glut.h>
#include <math.h>
#include <time.h>

float theta=30,thetar;
void wait(int timeout)
{
 timeout += clock();
 while(clock() < timeout)
  continue;
}
void idle()
{
    theta+=1; /* or some other amount */
    if(theta >= 360.0) theta-=360.0;
    wait(100);// control by the value according to system speed
    glutPostRedisplay();
}
void mouse(int button, int state, int x, int y)
{
    if(button==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
        glutIdleFunc(idle);
    if(button==GLUT_MIDDLE_BUTTON && state==GLUT_DOWN)
        glutIdleFunc(NULL);
}
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_POLYGON);
    /* convert degrees to radians */
    thetar =theta/(3.14159/180.0);
    glVertex2f(cos(thetar), sin(thetar));
    glVertex2f(-sin(thetar), cos(thetar));
    glVertex2f(-cos(thetar),-sin(thetar));
    glVertex2f(sin(thetar),-cos(thetar));
    glEnd();
    glFlush();
}
void myinit()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-2.0, 2.0, -2.0, 2.0);
    glMatrixMode(GL_MODELVIEW);
    glClearColor (1.0, 1.0, 1.0, 1.0);
    glColor3f(0.0,0.0,0.0);
}
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Rotating square");
    glutDisplayFunc(display);
    myinit();
    glutMouseFunc(mouse);
    glutIdleFunc(idle);
    glutMainLoop();
}
```

4.

- As long as the contents of the frame buffer are unchanged and we refresh at the 60- to 85-Hz rate, we should not notice the refresh
- When drawing a simple image that takes less than one refresh cycle timer (the drawing starts and ends in the same cycle) the image should be smooth
- When drawing a complex image that takes longer than one refresh cycle time, we se different parts each refresh cycle. Hence, artifacts may occur
- When clearing and redrawing a simple object (square for example) there is no coupling between when new squares are drawn into the frame buffer and when the frame buffer is

redisplayed by the hardware. Thus, depending on exactly when the frame buffer is displayed, only part of the square may be in this buffer. Hence, flickers and artifacts may appear.